

Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică

DOCUMENTAȚIA PROIECTULUI

HealthCare Application

STUDENȚI

Țupea Andrei Cristian

Popescu Andrei-Alexandru

Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică

Cuprins

1. Prezentarea Proiectului
2. Tehnologiile folosite
3. Backend-ul aplicatiei
4. Arhitectura aplicatiei
5. Domeniul de utilizare al aplicatiei
6. Concluzii

Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică

1. Prezentarea proiectului.

Aplicaţia noastră îşi propune să ajute sistemul medical atât din România cât şi din alte ţări care întâmpină dificultăţi în comunicarea dintre pacienţi şi personalul medical.

Proiectul nostru are ca scop limitarea nevoilor de deplasare şi aşteptări lungi la cozile din faţa spitalelor prin crearea unei conexiuni rapide şi simple între pacient şi medic. Prin aplicaţie te poţi conecta ca şi doctor sau ca şi pacient, în urma conectării vei fi redirecţionat către atribuţiile şi beneficiile specifice tipului de utilizator ales. De exemplu un medic poate uşor să vadă nu doar numele pacienţilor săi dar şi vârsta, înălţimea, greutatea, adresa de email prin care poate lua legătura cu acesta dacă este cazul. De asemenea doctorul poate simplu şi rapid să menţioneze diagnosticul oricărui pacient, să îi ofere acestuia o reţetă medicală digitală şi un sfat medical în legătură cu problema acestuia. Toate acestea se întâmplă în câteva secunde fara greutate şi fara timp pierdut din partea medicului sau a pacientului. Dacă alegem să ne logăm ca şi pacient putem alege în orice moment medicul care dorim să ne ofere tratament, aceştia sunt afişaţi într-o listă şi ne putem “abona” la oricare medic în orice moment. De asemenea reţetele şi sfaturile medicale trimise de medici pentru a putea începe tratamentul cât mai repede şi cât mai eficient.

1.2 Îmbunătăţiri

Din punct de vedere al îmbunătăţirilor, în momentul de faţă un pacient se poate “abona” sau “dezabona” de la un medic de cate ori

dorește în orice perioadă de timp, un mecanism de securitate pentru a preveni situațiile neplăcute. De asemenea medicamentele pot avea reacții

Universitatea Transilvania din Brașov

Facultatea de Matematică și Informatică

adverse în orice moment pentru diferite tipuri de persoane, ideea unui sistem prin care pacienții să poată să menționeze efectele de care suferă iar în timp dacă numărul de afecțiuni este crescut să se poată elimina acest medicament de pe piață.

De asemenea autentificarea este la un nivel de bază, îmbunătățiri precum autentificare cu amprentă, recuperare parole prin email sau telefon etc. sunt mai mult decât folositoare pentru o aplicație de acest tip.



Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică

2.Tehnologii folosite

2.1 Android studio IDE

Proiectul a fost creat în IDE-ul android studio, la momentul actual cel mai atrăgător şi cel mai performant pentru crearea aplicaţiilor mobile pentru sistemul de operare Android.

Android studio este bazat pe software-ul InteliIJ de la compania JetBrains.

2.2 Limbajul de programare ales

Pentru proiect am ales limbajul de programare Java, care este cunoscut pentru versatilitatea de care dă dovadă în orice situaţie. Pentru android studio pe lângă java există suport oficial şi pentru limbajul Kotlin, cu toate acestea am preferat java datorită faptului că suntem mai familiari cu el şi cel puţin la un nivel de bază îi cunoaştem performanţele.

2.3 Firebase

O aplicaţie de acest tip avea neapărat nevoie de o formă de autentificare, mai mult de atât, avea nevoie de o formă de autentificare sigură pentru că informaţii personale sunt folosite în domeniul aplicaţiei. Astfel am ales Firebase, acesta oferă o varietate de modalităţi de autentificare de reţineri de parole, de recuperări de parole, conferă securitate la nivel înalt, iar pe lângă securitate şi versatilitate oferă şi un panel “analytics” în care activitatea utilizatorilor pe aplicaţie este monitorizată în timp real.

Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică

3. Backend – request-uri folosite.

Aplicaţia nu a fost nevoită să opteze pentru un https request de tipul deserialize/parson JSON deoarece datele sunt în majoritate introduse de către utilizator şi apoi prelucrate fie în baza de date locală fie în API-ul Firebase.

Aşa că din acest motiv am creat un nou fragment de credits, l-a începutul aplicaţiei unde utilizatorul poate cere să primească numele creatorului şi email-ul acestuia.

Pentru a realiza acest lucru ne-am folosit de RequestQueue din librăria Volley, cu ajutorul website-ului mocki.io am reuşit să creem un fake api din care să putem realiza request-ul.

```
private void jsonParse(){
    String url = "https://mocki.io/v1/ecb79a00-1466-4210-a493-5bdaeef6db8";
    JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, jsonRequest: null,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try {
                    JSONArray jsonArray = response.getJSONArray( name: "creators");

                    for (int index = 0; index < jsonArray.length(); index++){
                        JSONObject creator = jsonArray.getJSONObject(index);

                        String name = creator.getString( name: "name");
                        String email = creator.getString( name: "email");

                        creditsText.append(name + "\n" + email + "\n\n");
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                // TODO Auto-generated method stub
            }
        })
    requestQueue.add(request);
}
```

4.Arhitectura aplicaţiei

4.1 Activităţi şi fragmente.

Aplicaţia este construită pe o singură activitate principală MainActivity şi o multitudine de fragmente, fiecare fragment având o singură utilitate. Aceste fragmente sunt încărcate pe rând în funcţie de nevoile utilizatorului, astfel avem:

- Fragmente de întâmpinare
- Fragmente de autentificare
- Fragmente principale pentru tipul de utilizator
- Fragmente auxiliare pentru tipul de utilizator

4.2 Navigare

Navigarea se face prin intermediul butoanelor sau în cazuri mai speciale prin intermediul imaginilor intuitive, care funcţionează pe post de butoane. Tehnologia din spate cu care se realizează tranziţia este FragmentTransaction. Cu ajutorul acesteia putem adăuga fragmente, le putem înlocui, sau şterge, pe lângă asta putem reveni uşor la fragmentele anterioare folosindu-ne de butonul “back” al telefonului, deoarece fragmentele vor rămâne în “backstack”. De asemenea prin această metodă se transmit date de la un fragment la altul unde este cazul.

4.3 Baza de date

Pentru stocarea datelor am decis să folosim baza de date locală Room. Am ales Room deoarece ne oferă aceeaşi putere ca şi altă bază de date precum SQLite, oferind un “layer” de abstractizare peste sqlite ca să ne confere acces fluent la datele stocate. Un alt avantaj major a fost verificarea în timpul compilării a Query-urilor şi a Entităţilor bazei de

date, cu ajutorul acestei verificări am putut ști în orice moment dacă sintaxa unui query era greșită, dacă în entități o anumită coloană nu exista etc. Pe lângă aceste verificări considerând că am ales să creem o bază de date relațională, în momentul compilării erau verificate și condițiile de primary key – foreign key dintre entități.

În general baza de date Room are 3 componente:

- Entity
- DAO
- Baza de date

Pe lângă cele 3 componente principale noi am folosit și Task-uri pentru fiecare query astfel încât să nu încărcăm main-threadul aplicației în timpul rulării. Aceste task-uri sunt apelate în aplicație prin intermediul unui Repository care poate chema oricare dintre task-uri în orice moment, în oricare dintre fragmentele noastre.

Simplitatea prin care Room tratează relațiile dintre tabele a fost de asemenea un factor decisiv în alegerea noastră, un query pentru a obține de exemplu toți pacienții care sunt abonați la un doctor, este aproape la fel de simplu ca și un query de select, deoarece avem acces la anotarea `@Relation` și `@Embedded` care fac accesul la datele relaționale simplu.

```
public class DoctorsWithPatients {  
    @Embedded public Doctor doctor;  
    @Relation(  
        parentColumn = "id",  
        entityColumn = "doctor_id"  
    )  
    public List<Patient> patients;  
}
```

```
@Transaction  
@Query("Select * From Doctor where id == :doctorId")  
List<DoctorsWithPatients> getDoctorWithPatients(Integer doctorId);
```


4.4 Recycler View

În majoritatea fragmentelor aplicației este nevoie de o listă care să conțină și să afișeze obiecte, fie pastile, fie pacienți etc. Pentru a rezolva această problemă într-un mod simplu ne-am folosit de Recycler view.

Cu ajutorul unui Adapter specific, se adaugă elementele în listă iar librăria Recycler view le crează și le afișează dinamic când este nevoie de ele. Cum ne sugerează și numele Recycler view “reciclează” elementele individuale din listă, când un element dispare de pe ecran din cauza scroll-ului, el nu este distrus, din potrivă se refolosește acel view pentru alte obiecte care apar pe ecran în momentul scroll-ului.

Recycler view a fost folosit în fiecare fragment principal, și în majoritatea fragmentelor auxiliare pentru eficiența și dinamicitatea de care dă dovadă.

4.5 Firebase

Librăria firebase a fost folosită în aplicație strict pentru autentificare. De asemenea utilizatorii autentificați sunt “trimiși” și în baza de date Room dar fără parola, aceasta este criptată de Firebase iar accesul la ea nu este permis.

Analizele în timp real pe care consola Firebase le oferă sunt de asemenea un plus adus aplicației pentru dezvoltări viitoare în funcție de utilizatorii care o vor folosi.



5. Domeniul de utilizare al aplicaţiei.

În momentul în care aplicaţia se deschide, utilizatorul este întâmpinat de un fragment simplu, cu 2 icon-uri reprezentând un doctor şi un pacient, utilizatorul poate alege din acest moment spre ce ramură aplicaţiei să se îndrepte. Indiferent că alege pacient sau doctor vă întâmpinat de un meniu din care vă alege logarea sau înregistrarea pentru tipul de utilizator selectat.

După autentificare utilizatorul va deschide un fragment de doctor sau de pacient în funcţie de ceea ce a ales la începutul aplicaţiei.

5.1 Doctor

În fragmentul doctorului vă găsi datele sale personale care pot fi updata-te cu ajutorul butonului de update (cu excepţia username-ului şi a email-ului).

Sub butonul de update se găseşte butonul de “Load patients” prin care lista de pacienţi abonaţi la acest doctor se vă încarca într-un recycler view dedesupt.

Doctorul poate da un click pe oricare dintre pacienţi pentru a deschide fragmentul auxiliar pentru doctor, în care poate scrie un diagnostic pentru pacientul ales, îi poate scrie un sfat şi să îi ofere reţetă medicala cu numele pastilei, indicaţii şi descriere, de menţionat faptul că el poate introduce toate aceste lucruri sau poate alege doar 1 sau doar două, şi poate reveni la orice moment în cazul unei modificări. Modificările vor apărea în fragmentul pacientului.

5.2 Pacient

În fragmentul pacientului vom găsi că și în cel al doctorului datele sale care pot fi modificate în mod asemănător cu ajutorul butonului de update. De menționat că pot fi modificate doar înălțimea, greutatea și vârsta (Diagnosticul va fi modificat de doctor, iar numele și email-ul nu pot fi modificate).

Sub datele personale avem un Recycler view care conține toți doctorii prezenți în baza de date, prin apăsarea butonului “subscribe” în colțul din dreapta al oricărui doctor acest pacient se poate abona la doctorul respectiv.

Desupra acestor datelor putem observa două butoane unul pentru Rețete respectiv pentru Sfaturile Doctorului. La un click la unul din aceste butoane putem găsi rețetele sau sfaturile doctorului care au fost adăugate anterior de doctor.



6. Concluzii

În concluzie aplicaţia HealthCare este utilizabilă şi are o utilitate ridicată mai ales în epoca vitezei în care eficienţa şi viteza este mai presus decât orice.

De menţionat este faptul că aplicaţia are nevoie în continuare de suport pentru a putea fi 100% viabilă pe piaţă.

- Are nevoie de o securitate mai ridicată
- Reţetele trebuie aprobate de sistemul medical astfel încât să poată fi acceptate în orice farmacie
- Doctorul trebuie să aibă un certificat de autenticitate care să poată fi vizibil pentru pacienţi
- Pacienţii trebuie să completeze un document online în momentul în care se abonează la un anumit doctor
- Reţetele trebuie să aibă un buton de “report” pentru a putea elimina de pe piaţă medicamente periculoase sau care nu mai sunt folosite

Dar considerând capacităţile aplicaţiei în momentul de faţă ar putea fi folosită de un grup mic de personal medical şi pacienţi care să poată oferi feedback în vederea îmbunătăţirii aplicaţiei înainte să fie încărcată în mediul online şi folosită de un număr mare de persoane.